[0051] The system verifier **164** performs one or more of the following verifications:

[0052] that all dependencies of installed software in the computer **102** are met

[0053] that an operating system of the computer **102** includes all device drivers necessary to run on the hardware configuration of the computer **102**;

[0054] that code and configuration settings in the computer **102** have not been altered either accidentally or maliciously;

[0055] that an application is correctly installed in the computer **102**;

[0056] that a known faulty or malicious program is not installed on the computer **102**;

[0057] that an application and all of its constituent components and dependencies exist on the computer **102** before installation;

[0058] that an application is installable on the computer **102** before loading its components onto a system;

[0059] that installation of a new application or system component will not conflict with existing applications or components;

[0060] that an application or operating system component can be removed without breaking dependencies from other applications or components;

[0061] that an application or operating system conforms to a predefined local policy;

[0062] that the necessary pre-conditions for application launch are met before the application is launched;

[0063] that applications for which pre-conditions are not met are not allowed to execute;

[0064] that during execution, if necessary conditions for execution are no longer valid, the application is no longer allowed to execute.

[0065] As shown in **FIG. 1**, another computer **170** is configured with its own memory **172** (e.g., volatile, non-volatile, removable, non-removable, etc.). This memory has a system inspector **180** therein.

[0066] As represented by the large-headed arrow, the system inspector **180** receives, as input, the "system image" of the computer **102**. In other words, it receives a copy of the system-embodying contents and configuration of the computer **102**. It may be received directly from the computer **102** or indirectly as a separate copy of the "system image."

[0067] The system inspector **180** performs an analysis of the offline "system image" to verify conclusively that the computer **102** contains specific functional components (such as the OS or applications). More particularly, the inspector examines the self-describing artifacts to see if all of the necessary components (described as such and referenced by manifests of the self-describing artifacts) are located and properly identified. The inspector reports the results of this examination.

[0068] The information contained in the manifest for an artifact can be used by a compiler or other optimization tool to facilitate the optimization of the code in the artifact, at install time, program load time, or another time of a user's choosing, by describing all of the libraries, components, and dependencies of the artifact. This description permits the compiler or tool to make more precise assumptions about the environment in which the artifact executes and the code within the artifact.

[0069] The information contained in the manifest for an artifact can be used by an error detection tool to facilitate ensuring the correctness of the code in the artifact, at install time, program load time, or another time of a user's choosing, by describing all of the libraries, components, and dependencies of the artifact. This description permits the tool to make more precise assumptions about the environment in which the artifact executes and the code within the artifact.

Manifest

[0070] A manifest contains metadata that describes artifacts of the computer **102**. The metadata also describes configuration information related to the artifacts including external dependencies and external interfaces. The manifests also describe the connectivity relationships between software components.

[0071] For example, the manifest for an application called "ProgramA", as delivered by its publisher, includes a list of the binary load modules (EXEs, DLLs, etc.), certificates attesting to the authenticity of the components and provider, a list of the names of configuration settings and their default values, a list of external binary load modules required by the program's load modules, a list of external settings and names accessed by the program, and a list of names and settings exposed by ProgramA, such as the information required to tell the operating system that ProgramA wants to be the default editor for files with the ".ZZZ" extension.

[0072] In at least one embodiment, the manifests of self-describing artifacts contain declarative descriptions that provide sufficient information to enable the following:

[0073] program-installation software to install or uninstall its program components without executing any ad-hoc code contained within the program;

[0074] the person or agent installing a program to bind its program components

[0075] the person or agent installing a program to override any configurable default settings;

[0076] the person or agent installing a program to make it part of the manifest for the system as a whole;

[0077] inspector software (e.g., system inspector **180**) to verify that a particular program has been correctly installed or uninstalled, both in its own context and in the context of the system as a whole;

[0078] verification software (e.g., system verifier **164**) to determine if a particular program is presently runnable;

[0079] pre-installation software to determine if all dependencies are met on a system necessary to enable a particular program to be installed and run;

[0080] the person or agent installing a particular program to determine other aspects of the future behavior of the program or of the system as a whole;